Determining a proper initial configuration of Red-Black planning by machine learning

Otakar Trunda and **Roman Barták** [1]

1 INTRODUCTION

Planning deals with finding a sequence of actions that transforms the world from a given initial state to a state that satisfies a certain goal condition [8]. For the purposes of this paper we can define a planning problem simply as a state-transition system where states are the world states and transitions correspond to application of actions. States are defined by values of state variables. Let X be the set of state variables, each variable x_i has a finite domain D_i of its possible values. Then state s is a mapping from X to $\bigcup_i D_i$. $s : X \mapsto \bigcup_i D_i$, where $\forall i, s(x_i) \in D_i$.

The state space has a form of the Cartesian product of variables' domains. $Space = \prod D_i$ Every state $s \in Space$ has assigned a (possibly empty) set of its successor states designed $succ(s)$, every $t \in succ(s)$ is labeled by the action that transforms s to t (i.e. performing actions changes values of state variables). The task is to find a path p in this state-transition system that leads from a given initial state to some state satisfying a goal condition (a goal state). $p = \{s_0, s_1, \ldots, s_n\}$, where s_0 is the initial state, s_n is some goal state and $\forall 0 \leq i < n : s_{i+1} \in succ(s_i)$. Such a path is called a *solution plan*. The goal is to reach a state where some variables have specified values.

One of the most promising approaches to solve the planning problem (based on the results of several International Planning Competitions [2]) is heuristic-guided forward search. (Mostly in a form of A^* or a hill-climbing). These approaches make use of a heuristic estimation during search and the accuracy of the heuristic estimator has a great impact on the performance. Hence designing a powerful and easy-to-compute heuristic is of paramount importance.

Heuristics are usually based on relaxations of the problem. When estimating the quality of the best solution, we relax the problem by ignoring some constraints (making the problem easier), then solve the relaxed problem and use the quality of that solution as a lower bound on the quality of the best solution to the original problem. In planning, this principle is represented by the well known *delete relaxation heuristic* and its variants [8, 3, 4]. Heuristics based on this principle often work well, but in some situations they greatly underestimate the real value making them inaccurate (see [6] for example).

Delete relaxation allows the state variables to hold several values simultaneously, so the *relaxed state* subsumes several ordinary states. Furthermore, performing actions (i.e. making transitions) only adds new elements to the set of values that each variable currently holds (never removes any value). Hence the set of ordinary states that the relaxed state subsumes monotonically increases on every path. A path is a *relaxed solution plan* if it leads to a relaxed state which subsumes some goal state. The length of relaxed plan is then used to estimate the length of the real plan.

[1] Charles University in Prague, Faculty of Mathematics and Physics, email: otakar.trunda@mff.cuni.cz, roman.bartak@mff.cuni.cz

2 RED-BLACK PLANNING

Red-Black planning is a new approach to heuristics design which generalizes the delete relaxation and compensates for many of its shortcomings with a reasonable computational effort [7, 6, 5]. It divides the set of state variables into two disjoint subsets - *Red* and *Black*, which are treated differently during the planning. The *Red* variables are treated as in the *delete relaxation* while the *Black* variables are not relaxed. If all variables are *Red* then the heuristic works same as the delete relaxation.

The authors showed that by the proper selection of *Red* variables, we can reduce the underestimation (in most cases) and still keep the method polynomial. They also observed that the selection of *Red* variables has a great impact on the overall performance. While proper selection leads to good performance, with poor selection the performance degrades. Selecting the proper variables, however, appears to be a hard problem.

The authors performed several tests with *intuitive* and *counter-intuitive* variable selection methods (where *intuitive* relaxes the least important variables, while the *counter-intuitive* method relaxes the most important variables). It turned out that the *counter-intuitive* method often beats the *intuitive* one (with respect to the time required for solving the problem) which makes the problem quite unpredictable. This led the authors to hypothesize that no *simple and efficient* method for selecting the variables can be found.

3 OUR METHOD

We believe that different domains require different ways of selecting the variables. We propose a method based on machine learning that works as follows: first it creates a set of small sub-problems of the original problem and then it determines the proper variable selection for these sub-problems (by enumerating all possibilities). Finally, it uses the solutions of sub-problems to derive the solution to the original problem.

3.1 Creating samples

We create the sub-problems by selecting small subsets of variables and restricting the original problem to these variables only. The restriction has a form of projection which preserves the paths - i.e. if there is a path from s to t in the original state-transition system, then there is a path from $restriction(s)$ to $restriction(t)$ in the new system. Of course, new paths may emerge during the restriction that were not present before.

Formally, let $\mathcal{A}$ be a planning problem as defined earlier, X its state variables, and $Space = \prod D_i$ its state space. Then for every $P \subseteq X$ called *pattern* and every state $s \in Space$ we define a *restriction* of s to P as $s^P : P \mapsto \bigcup D_i$, where $\forall x_i \in P, s^P(x_i) = s(x_i)$. A *restriction* of $\mathcal{A}$ to a pattern P is a planning problem $\mathcal{A}^P$ with state variables P, state space $Space^P = \prod_{\{i|x_i \in P\}} D_i$, and for $s, t \in Space^P : s \in succ(t)$ if and only if there exist $u, v \in Space$ such that $s = u^P, t = v^P$ and $u \in succ(v)$. The initial state and goal states of the restricted problem are restrictions of the originals.

In each sub-problem induced by a pattern, we create samples by enumerating all ways of selecting the Red variables. A *sample* then consists of a pair *(pattern, selected Red variables)*.

3.2 Evaluating samples

Let X be the set of state variables of the original planning problem. A sample q is given in a form $q = (P_q, R_q)$, where P_q is the pattern and R_q is the set of selected red variables, $R_q \subseteq P_q \subseteq X$. To evaluate the sample q, we chose the following procedure:

1. Restrict the original problem to the pattern P_q
2. Solve the restricted problem by A^* with the Red-Black heuristic using R_q as a set of red variables.
3. Measure the time required to perform step 2 in seconds and use it to evaluate the sample. ($Val(q)$ denotes the value of a sample q.)

We decided to use the run-time to evaluate the sample rather than other characteristics like heuristic calls or expanded nodes. We believe that using such characteristics would bias the selection in favor of large patterns and small *Red* sets, since such combination would lead to a very accurate heuristic. However, such heuristic might take a long time to compute and probably wouldn't be the best alternative.

Since run-time of the whole process is the criterion we want to optimize, it seems appropriate to use it to evaluate samples.

3.3 Learning from samples

After evaluating enough samples, we have to select the red variables for the original problem. In our preliminary experiments, we used the following simple procedure, but we believe that this phase can yet be perfected by using more sophisticated approach.

1. Given the set of samples Q, a sample $q = (P_q, R_q)$, divide the samples to groups by the pattern they use. $Q_P = \{q \in Q \mid P_q = P\}$
2. Select the best sample in each group Q_P (one with the lowest evaluation), and denote its Red set as $Best^P$.
3. For each state variable count how many times it appears in some $Best$ set. $val(x_i) = \left|\{Best^P \mid x_i \in Best^P\}\right|$
4. Select variables with the highest evaluation.

In step four, the number of variables to select can be a fixed constant or a fixed ratio, but we chose a different approach. Suppose there are n state variables. We sort the variables nonincreasingly by their evaluation: $\{x_1, x_2, \ldots, x_n\}$, where $val(x_i) \geq val(x_{i+1})$. We add the first variable and then keep adding more until $val(x_i) - val(x_{i+1}) > \frac{val(x_1) - val(x_n)}{n}$. This stopping criterion should find the gap between the *good* variables and the *bad* variables. We intend to test other selection policies as well.

Step three can be generalized by introducing weights to the *Best* sets. Currently, each *Best* set has a weight of 1, but larger patterns give us more information since they are closer to the original problem. Step three can be modified to $val(x_i) = \sum_{\{Best|x_i \in Best\}} w(Best)$, where w is a weight function. We used $w(Best) = |Best|$, but different functions are also possible.

Finally, step two can be modified to work with more samples than just the best one. Imagine that there might be a variable which is rarely in the best sample, but often in the second best one. This would still be a good candidate to pick. In step three we would then average the evaluation of all samples that contain the variable x_i, possibly weighted according to the size of the pattern they use. This modification should lead to more accurate results, but it takes more time to compute. Therefore it is not yet clear whether or not it will improve the overall performance.

4 CONCLUSIONS AND FUTURE WORK

We present the parameter learning method in a very simple form, many issues remain unresolved. Preliminary experiments show promising results, but the method still needs to be adjusted and properly tested on a larger set of planning domains.

One part we didn't address yet is the selection of patterns during the creation of samples. Unlike typical machine learning applications, here we can decide what samples we use for the learning. Patterns should be selected iteratively and the selection should be based on previous results and should support both exploration and exploitation. We intend to make use of some Monte-Carlo technique, possibly Monte-Carlo Tree Search [1]. The method is guaranteed to converge to optimal solution if patterns are chosen incrementally (as the size of the pattern grows, the sub-problem converges to the original problem). The speed of convergence, however, needs yet to be determined for various domains.

The proposed method of learning from pattern-induced subproblems is not bound to the Red-Black planning heuristic only, but can be used to gain information about other features of the planning problem as well. Such information might then help to improve various search methods.

ACKNOWLEDGEMENTS

The research is supported by the Grant Agency of Charles University under contract no. 390214 and it is also supported by SVV project number 260 104.

REFERENCES

[1] C.B. Browne et al., 'A survey of monte carlo tree search methods', *Computational Intelligence and AI in Games, IEEE Transactions on*, **4**(1), 1–43, (March 2012).
[2] ICAPS Competitions. http://ipc.icaps-conference.org, June 2014.
[3] Jörg Hoffmann, 'Where "ignoring delete lists" works: Local search topology in planning benchmarks', *J. Artif. Int. Res.*, **24**(1), 685–758, (November 2005).
[4] Jörg Hoffmann, 'Where Ignoring Delete Lists Works, Part II: Causal Graphs', in *21st International Conference on Automated Planning and Scheduling*, Freiburg, Allemagne, (2011).
[5] Michael Katz and Jörg Hoffmann, 'Red-black relaxed plan heuristics reloaded.', in *SOCS*, eds., Malte Helmert and Gabriele Rger. AAAI Press, (2013).
[6] Michael Katz, Jörg Hoffmann, and Carmel Domshlak, 'Red-black relaxed plan heuristics', in *AAAI'13*, (2013).
[7] Michael Katz, Jörg Hoffmann, and Carmel Domshlak. Who said we need to relax all variables?, 2013.
[8] Dana Nau, Malik Ghallab, and Paolo Traverso, *Automated Planning: Theory & Practice*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.